

Optimal Nonparametric Estimation of First-Price Auctions

BY EMMANUEL GUERRE, ISABELLE PERRIGNE AND QUANG VUONG (2000)

Edmund Y. Lou

Abstract

The only purpose of this report is to replicate the two-step nonparametric estimation proposed by Emmanuel Guerre, Isabelle Perrigne and Quang Vuong (2000). The abbreviation GPV is used throughout the text.

1 Introduction

With auction data increasingly available from different sources, such as government procurement, eBay and Google page positioning, the modern industrial economics is facing a challenge of its empirical usefulness. Having this in mind, GPV proposes a two-step nonparametric procedure to estimate the density of private values in the first-price sealed-bid auction under the independent private value paradigm.

GPV successfully answers three fundamental questions associated with structural analysis: (i) Whether a theoretical model places restrictions on observed data to be testable, (ii) Whether a structural analysis requires a priori parametric information, and (iii) Whether an estimation procedure can be free from parametric assumptions.

Our aim is to replicate such two-step nonparametric method in the following sections.

2 The First-Price Sealed-Bid Auction Model

Consider an auction consists of I risk-neutral players, who bid for a single and indivisible object. Each player i knows privately how much the object is worth to her (i.e., her private value v_i for the object) but does not know other players' valuations. Each player considers the private values of the object to the other $I - 1$ players be independent random variables drawn from an interval $[\underline{v}, \bar{v}] \subset \mathbb{R}_+$ with a common absolutely continuous distribution F . In the auction, each player i simultaneously submits a sealed bid b_i and the object is sold to the player whose bid is the highest. The player who wins the auction must pay the amount of her bid, provided that the bid is no lower than the reserve price p_0 of the owner of the object, while the other players pay nothing. The distribution F with its density f ,

the number of players I , and the reserve price $p_0 \in [\underline{v}, \bar{v}]$ are common knowledge among all the players.

This auction model is in nature a Bayesian game where each player's type is given by her private value for the object. The corresponding (symmetric) Bayesian Nash equilibrium bid b_i of the i -th player, assuming $I \geq 2$, is

$$(1) \quad b_i = s(v_i; F, I, p_0) = v_i - \frac{1}{(F(v_i))^{I-1}} \int_{p_0}^{v_i} (F(u))^{I-1} du,$$

whenever $v_i \geq p_0$. Note that $s(\cdot; F, I, p_0)$ is strictly increasing and that b_i can be any value strictly less than p_0 such that $v_i < p_0$. The equilibrium bid is obtained from maximising expected payoff

$$\max_{b_i} (v_i - b_i) F(s^{-1}(b_i))^{I-1},$$

namely, from solving the (rearranged) first-order condition

$$(2) \quad s'(v_i) = [v_i - s(v_i)](I-1) \frac{f(v_i)}{F(v_i)}.$$

with boundary condition $s(p_0) = p_0$.

Let G be the distribution of b_i . Due to the strict monotonicity,

$$G(b_i) = \mathbb{P}(b \leq b_i) = \mathbb{P}(v \leq s^{-1}(b_i)) = F(v_i), \quad b_i \in [\underline{v}, s(\bar{v})].$$

It follows that $f(v_i) = g(b_i)s'(v_i)$, where g is the density function of b_i . Taking the ratio gives that

$$\frac{g(b_i)}{G(b_i)} = \frac{f(v_i)}{F(v_i)} \frac{1}{s'(v_i)}.$$

Thus (2) can be rewritten as

$$(2') \quad v_i = \xi(b_i; G, I) = b_i + \frac{g(b_i)}{(I-1)G(b_i)}.$$

Equation (2') is the linchpin to the nonparametric estimation discussed in the next section.

3 Identification and Estimation

In auction bids are observed but private values are commonly unobservable. Whenever we know the distribution function G and its density g , the private values can be easily recovered from equation (2'). However, they are unknown in general. Thanks to the modern development of econometric techniques, they can be estimated nonparametrically. This suggests the following two-step estimation.

First, using observed bids we estimate G and g by empirical distribution and kernel density estimator, respectively; that is, by

$$(3) \quad \tilde{G}(b) = \frac{1}{IL} \sum_{l=1}^L \sum_{p=1}^I \mathbf{1}(B_{pl} \leq b),$$

and

$$(4) \quad \tilde{g}(b) = \frac{1}{ILh_g} \sum_{l=1}^L \sum_{p=1}^I K_g \left(\frac{b - B_{pl}}{h_g} \right),$$

where K_g is a kernel with a compact support, denoted by ρ_g its length, and h_g is a bandwidth.

Second, let B_{\min} and B_{\max} be the minimum and maximum of the observed bids and assume that there are L auctions. Since \tilde{g} is asymptotically unbiased on $[B_{\min} + \rho_g h_g/2, B_{\max} - \rho_g h_g/2]$, we can define the pseudo private value \hat{V}_{pl} corresponding to its bids B_{pl} by

$$(5) \quad \hat{V}_{pl} = \begin{cases} B_{pl} + \frac{1}{I-1} \frac{\tilde{G}(B_{pl})}{\tilde{g}(B_{pl})} & \text{if } B_{\min} + \frac{\rho_g h_g}{2} \leq B_{pl} \leq B_{\max} - \frac{\rho_g h_g}{2} \\ +\infty & \text{otherwise} \end{cases},$$

where $p = 1, \dots, I$ and $l = 1, \dots, L$. The set of pseudo private values is finally utilised to estimate the density function of the private values via

$$(6) \quad \hat{f}(v) = \frac{1}{ILh_f} \sum_{l=1}^L \sum_{p=1}^I K_g \left(\frac{v - \hat{V}_{pl}}{h_f} \right).$$

Here K_f and h_f again are a kernel and a bandwidth, respectively.

4 Monte Carlo Experiments

To illustrate the two-step nonparametric procedure, we conduct a Monte Carlo experiment with 1000 replications. Every one of our replications consists of $L = 200$ auctions, each having $I = 5$ players. The true distribution F of private values is assumed to be log-normal with parameters zero and one, truncated at 0.055 and 2.5. Its corresponding distribution function is the following:

$$-0.00227682 + 0.610964 \times \operatorname{erfc} \left[\frac{\log(x)}{\sqrt{2}} \right], \quad x \in [0.055, 2.5],$$

where $\operatorname{erfc}(\cdot)$ is the complementary error function.

Unlike the program FORTRAN used in GPV, we use Julia instead. Then we apply the following simulation and estimation procedure for each replication.

- Step 1: Simulate $I \times L$ private values from F .
- Step 2: Compute numerically the corresponding “observed” bids B_{pl} using (1) with reserve price $p_0 = 0.055$.

- Step 3: Estimate the distribution function and density function of observed bids using (3) and (4). The kernel is chosen to be the triweight kernel

$$K_g(u) = \frac{35}{32}(1 - u^2)^3 \mathbf{1}(|u| \leq 1).$$

Also, the bandwidth h_g is either selected from the set $\{0.01 \times n: n = 1, \dots, 10\}$ or chosen by the so-called Silverman's rule of thumb, namely, $h_g = 1.06\hat{\sigma}_b(IL)^{-1/5}$, where $\hat{\sigma}_b$ is the estimated standard deviation of the observed bids B_{pl} .

- Step 4: Calculate the pseudo private values \hat{V}_{pl} using (5). Since the kernel K_g is triweight, we have $\rho_g = 2$.
- Step 5: Estimate private value density function \hat{f} using (6). Here K_f and h_f are chosen by the same way as mentioned in Step 3.

Note that each replication, evaluated at 500 equally spaced points on $[b, \bar{b}] = [0.055, s(2.5)]$ and $[0.055, 2.5]$, gives two estimated functions $\hat{\xi}$ of (2') and \hat{f} of (6). Our Monte Carlo results are summarised in Figures 1–4.

Figure 1 illustrates the true equilibrium bid and the mean, the 5% percentile and 95% percentile of the 1000 estimates of $\hat{\xi}(B_{pl})$ against equally spaced private values. The two percentiles give the estimated 90% confidence interval for $\xi(b_i)$. Then we observe that the mean excellently matches the true values while falls within the confidence interval if it is bordered by the two horizontal lines defined by $B_{\min} + h_g$ and $B_{\max} - h_g$.

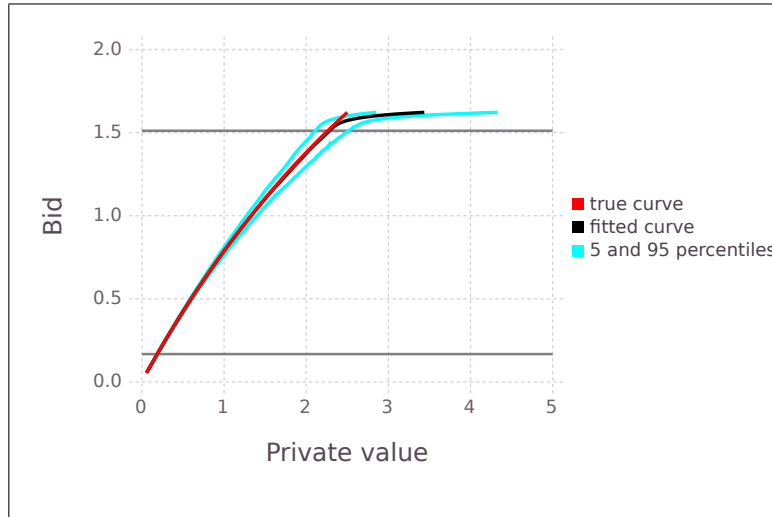


Figure 1: True and estimated equilibrium bids

Figure 2–4 demonstrate the true density and the mean, the 5% percentile and 95% percentile of the 1000 estimates of $\hat{f}(\hat{V}_{pl})$ against equally spaced private values. Similarly, the two percentiles give the estimated 90% confidence interval for $f(v_i)$ and the two vertical lines are given by $\hat{\xi}(B_{\min} + h_g) + h_f$ and $\hat{\xi}(B_{\max} - h_g) - h_f$. Both bandwidths h_g

and h_f are chosen by the rule of thumb in Figure 2, with the latter trimmed corresponding to (5). The mean matches well the true density within the range bordered by two vertical bars.

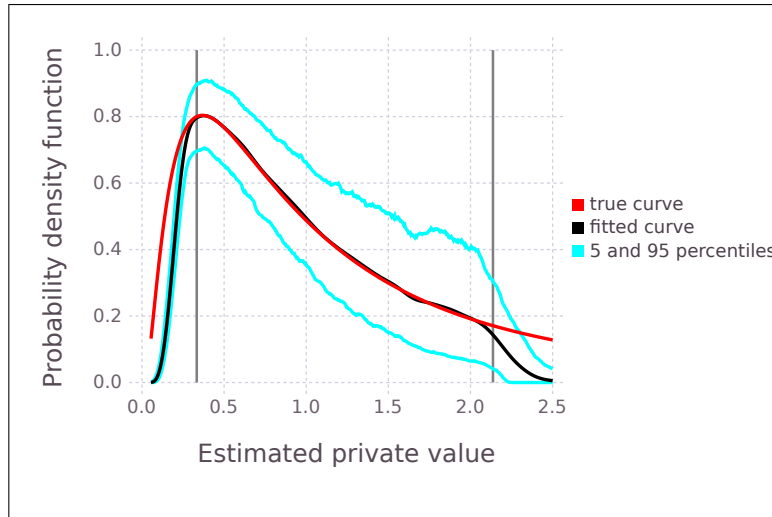


Figure 2: True and estimated density of privat values

Figure 3 is drawn with h_f fixed to be rule-of-thumb and $h_g = 0.02$. We can see that in this figure the mean matches better the true density than in Figure 2 because only the right vertical bar is binding.

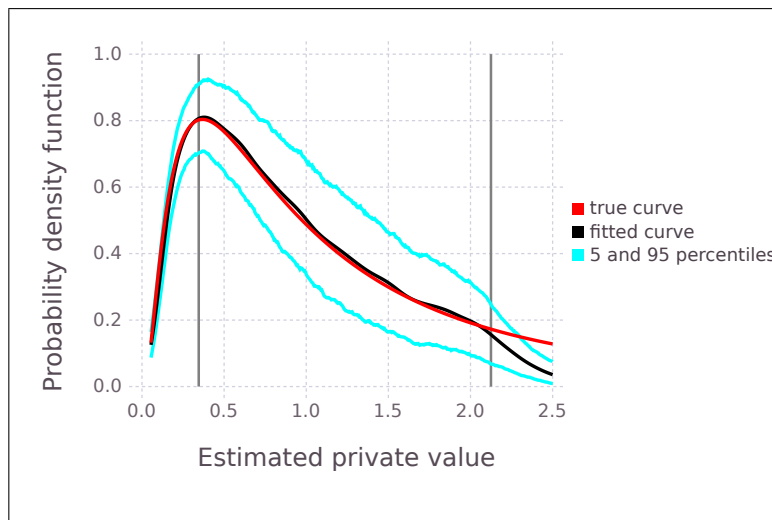


Figure 3: True and estimated density of privat values

Figure 4 is plotted with $h_g = h_f = 0.02$. The mean matches perfectly the true density in a bigger range but with wider confidence band.

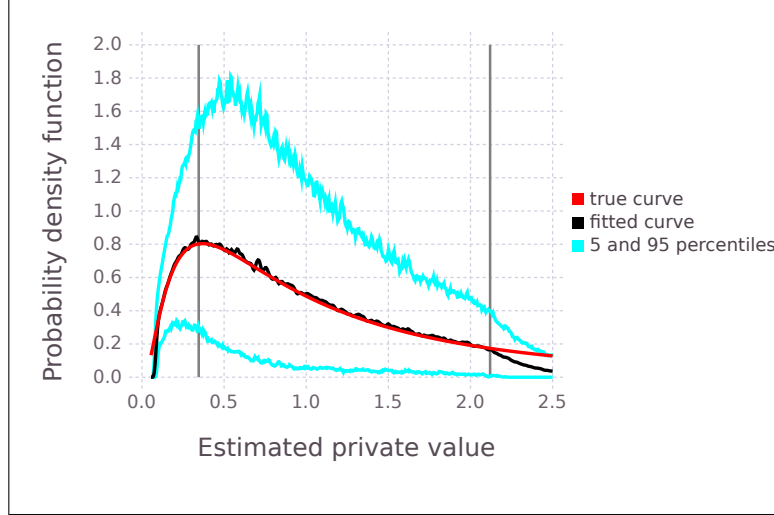


Figure 4: True and estimated density of privat values

5 Conclusion

To sum up, the two-step nonparametric method proposed in GPV is in principle general because it can be generalised to any other auction model. For example, if we only observe winning bids, the same idea still applies with one more piece added: the relationship between the distribution of winning bids and the distribution G . Within the independent private value paradigm, the former is $G(\cdot)^I$, namely, the first of the order statistic. With the avoidance of equilibrium bid determination, the computation is substantially simplified as we do not need to solve numerically the differential equation (2). Last but not least, GPV answers that the restrictions imposed by such a theoretical auction model can form a basis of a test of the theory per se.

A Julia Code

```
using StatsBase
using Gadfly
using Distributions
using Cairo

I = 5      # number of bidders for each replication
L = 200    # number of auctions for each replication

## simulate 1000 * IL = 1000 * 1000 private values from a truncated
## log-normal(0, 1) distribution
private_values = zeros(1000, I*L)
for i = 1:1000
    private_values[i, :] = rand(Truncated(LogNormal(0, 1), 0.055, 2.5), I*L)
end
```

```

function Bid(xmat)
# =====
# Purpose: Determine equilibrium bid from
#           its corresponding private value
# Input:   xmat = private value, a matrix
# Output:  equilibrium bids
# =====
xsize = size(xmat)
nrow = xsize[1]
ncol = xsize[2]
out = zeros(xsize)
tunc(x) = -0.00227682 + 0.610964.* erfc(-log(x)./sqrt(2))
tunc4(x) = tunc(x).^(I - 1)
for i = 1:nrow
    for j = 1:ncol
        step1 = (tunc(xmat[i, j])).^(I - 1)
        step2 = quadgk(tunc4, 0.055, xmat[i, j])[1]
        out[i, j] = xmat[i, j] - step2 ./ step1
    end
end
return out
end

```

```

bids = Bid(private_values)
bound = Bid([0.055 2.5])

```

```

function myindicator(xvec, bound)
# =====
# Purpose: Define the indicator function
#           of the form  $1(|xvec| \leq bound)$ ;
#           xvec is a vector
# =====
nx = length(xvec)
for i=1:nx
    if abs(xvec[i]) <= bound
        xvec[i] = 1
    else
        xvec[i] = 0
    end
end
return xvec
end

```

```

function silverman(xvec)

```

```

# =====
# Purpose: Calculate bandwidth using
#           Silverman's rule of thumb
# Input:   xvec = a vector of
#           private values
# =====

sd_hat = std(xvec)
nx = length(xvec)
out = (4.0 .* (sd_hat.^5))./(3.0 * nx)
h = out.^(1/5)
return h
end

function epa(xvec)
# =====
# Purpose: Define Epanechnikov kernel
# =====
out = (3/4).*(1-xvec.^2).* myindicator(xvec, 1.0)
return out
end

function triweight(xvec)
# =====
# Purpose: Define triweight kernel
# =====
out = (35/32).*((1-xvec.^2).^3).* myindicator(xvec, 1.0)
return out
end

function gfunc(xvec, data, func, h)
# =====
# Purpose: Define kernel density estimator
#           corresponding to g(.) in GPV
# Input:   xvec = input values, a vector
#           data = equilibrium bids
#           func = kernel function
#           h = bandwidth
# =====
nx = length(xvec)
nd = length(data)
coeff = 1/(nd.* h)
out = zeros(nx)
for i = 1:nx
    out[i] = coeff.* sum(func((xvec[i] - data)./h))
end

```



```

    return out
end

```

```

function Gfunc(xvec, data)
# =====
# Purpose: Define empirical distribution
#           corresponding to G(.) in GPV
# Input:   xvec = input values, a vector
#           data = equilibrium bids
# =====
nx = length(xvec)
nd = length(data)
out = zeros(nx)
for i = 1:nx
    outt = zeros(nd)
    for j = 1:nd
        if data[j] <= xvec[i]
            outt[j] = 1
        else
            outt[j] = 0
        end
    end
    out[i] = sum(outt)./nd
end
return out
end

```

```

function Vhat(xvec, data, func, h)
# =====
# Purpose: Calculate trimmed pseudo
#           private value
# Input:   data = equilibrium bids
#           func = kernel function
#           h = bandwidth
# =====
nx = length(xvec)
xmin = minimum(xvec)
xmax = maximum(xvec)
out = NaN
for i = 1:nx
    if (xmin + h) <= xvec[i] <= (xmax - h)
        temp = xvec[i] + (1./(I - 1)) .* (Gfunc(xvec[i], data)./gfunc(xvec[i], data, func, h))
        out = [out, temp]
    end
end
end

```

```

    return out[2: end]
end

function xihat(xvec, data, func, h)
# =====
# Purpose: Calculate estimated private
#          values
# Input:   xvec = input values, a vector
#          data = equilibrium bids
#          func = kernel function
#          h = bandwidth
# =====
    out = xvec + (1./(I - 1)) .* (Gfunc(xvec, data)./gfunc(xvec, data, func, h))
end

#####
## producing Figure 1
#####
bound = Bid([0.055 2.5])
step_v = linspace(0.055, 2.5, 500)
step_b = vec(Bid(reshape(step_v, 1, 500)))
value_hat = zeros(1000, 500)
hline_low = zeros(1000)
hline_up = zeros(1000)

for i = 1:1000
    temp = silverman(bids[i, :])
    value_hat[i, :] = xihat(step_b, bids[i, :], triweight, temp)
    hline_up[i] = maximum(bids[i, :]) - temp
    hline_low[i] = minimum(bids[i, :]) + temp
end

hline_up = fill(mean(hline_up), 1000)
hline_low = fill(mean(hline_low), 1000)
vmean = zeros(500)
v5 = zeros(500)
v95 = zeros(500)

for i = 1:500
    vmean[i] = mean(value_hat[:, i])
    v5[i] = percentile(value_hat[:, i], 5)
    v95[i] = percentile(value_hat[:, i], 95)
end

layer1 = layer(x = step_v, y = step_b, Geom.line, Theme(default_color = color("red")),

```

```

        line_width = 2px))
layer2 = layer(x = vmean, y = step_b, Geom.line, Theme(default_color = color("black"),
        line_width = 2px))
layer3 = layer(x = v5, y = step_b, Geom.line, Theme(default_color = color("cyan"),
        line_width = 2px))
layer4 = layer(x = v95, y = step_b, Geom.line, Theme(default_color = color("cyan"),
        line_width = 2px))
layer5 = layer(x = 0:5, y = hline_up, Geom.line, Theme(default_color = color("grey"),
        line_width = 1.5px))
layer6 = layer(x = 0:5, y = hline_low, Geom.line, Theme(default_color = color("grey"),
        line_width = 1.5px))
plot(layer1, layer2, layer3, layer4, layer5, layer6, Guide.xlabel("Private value"),
        Guide.ylabel("Bid"), Guide.manual_color_key("",
        ["true curve", "fitted curve", "5 and 95 percentiles"],
        [color("red"), color("black"), color("cyan")]))

#####
## producing Figure 2-4
#####
step_v = linspace(0.055, 2.5, 500)
fhat = zeros(1000, 500)
vline_up = zeros(1000)
vline_low = zeros(1000)

for i = 1:1000
    hg = 0.02
    hf = silverman(pseudo)
    pseudo = Vhat(bids[i, :], bids[i, :], triweight, hg)
    fhat[i, :] = gfunc(step_v, pseudo, triweight, hf)
    temp1 = silverman(bids[i, :])
    temp2 = hf
    vec1 = [minimum(bids[i, :]) + temp1]
    vec2 = [maximum(bids[i, :]) - temp1]
    vline_low[i] = xihat(vec1, bids[i, :], triweight, temp1)[1] + temp2
    vline_up[i] = xihat(vec2, bids[i, :], triweight, temp1)[1] - temp2
end

fhat_mean = zeros(500)
fhat_5perc = zeros(500)
fhat_95perc = zeros(500)

for i = 1:500
    fhat_mean[i] = mean(fhat[:, i])
    fhat_5perc[i] = percentile(fhat[:, i], 5)

```

```

    fhat_95perc[i] = percentile(fhat[:, i], 95)
end

vline_up = fill(mean(vline_up), 1000)
vline_low = fill(mean(vline_low), 1000)

ticks = linspace(0.0, 2.0, 11)
y1 = pdf(Truncated(LogNormal(0, 1), 0.055, 2.5), step_v)

layer1 = layer(x = step_v, y = y1, Geom.line, Theme(default_color = color("red"),
    line_width = 2px))
layer2 = layer(x = step_v, y = fhat_mean, Geom.line, Theme(default_color = color("black"),
    line_width = 2px))
layer3 = layer(x = step_v, y = fhat_5perc, Geom.line, Theme(default_color = color("cyan"),
    line_width = 2px))
layer4 = layer(x = step_v, y = fhat_95perc, Geom.line, Theme(default_color = color("cyan"),
    line_width = 2px))
layer5 = layer(x = vline_up, y = 0:2, Geom.line, Theme(default_color = color("grey"),
    line_width = 1.5px))
layer6 = layer(x = vline_low, y = 0:2, Geom.line, Theme(default_color = color("grey"),
    line_width = 1.5px))
plot(layer1, layer2, layer3, layer4, layer5, layer6, Guide.xlabel("Estimated private value"),
    Guide.ylabel("Probability density function"), Guide.yticks(ticks = ticks),
    Guide.manual_color_key("", ["true curve", "fitted curve", "5 and 95 percentiles"],
    [color("red"), color("black"), color("cyan"), ]))

```

References

- [1] Guerre, E., I. Perrigne, and Q. Vuong (2000): "Optimal Nonparametric Estimation of First-Price Auctions," *Econometrica*, 68, 525-74.